Project 1: Path Finding A*, UCS, BFS, DFS

Nour Eddine Amraoui

Dr. Tajjeedine Rachidi

Intro. to Artificial Intelligence

Feb 13th, 2022

Tabel of Contents

The project objective

The main objective of this project is to visualize and compare different algorithms for path finding. The software used to perform the stated tasks is unity.

Link to the project on Github

https://github.com/nouredd88/PathFinding_Project1

Functions used in the project

- FindPathA: finds shortest path using A* with the video heuristic.
- FindPathAEuc: finds shortest path using A*and the Euclidean heuristic.
- FindPathAMan: finds shortest path using A*and the Manhattan heuristic.
- FindPathUCS: finds shortest path using uniform cost search.
- FindPathBFS: finds shortest path using Breadth first search.
- FindPathDFS: finds shortest path using Depth first search.

- RetracePathA, RetracePathAEuc, RetracePathAMan Keep track of nodes that constitute the shortest path using A* depending on the corresponding heuristic.
- RetracePathUCS: Keep track of nodes that constitute the shortest path using UCS.
- RetracePathBFS: Keep track of nodes that constitute the shortest path using BFS.
- RetracePathDFS: Keep track of nodes that constitute the shortest path using DFS.

- GetDistance: Video heuristic gets the optimal distance between the two parameters of the function considering that some directions have a higher cost that normal movement
- GetDistanceSQRT: Euclidean heuristic.
- GetDistanceManhattan: Manhattan heuristic.

- File containing colorization and grid structure: Grid2.cs

- File containing the node structure: Node.cs

The Analysis

The algorithms that are implemented in this project along with their corresponding colors in the grid are the following:

- DFS: Blue
- BFS: Yellow
- UCS: Green
- A* (video heuristic): Cyan
- A* (Manhattan heuristic): Grey
- A* (Euclidean heuristic): Magenta

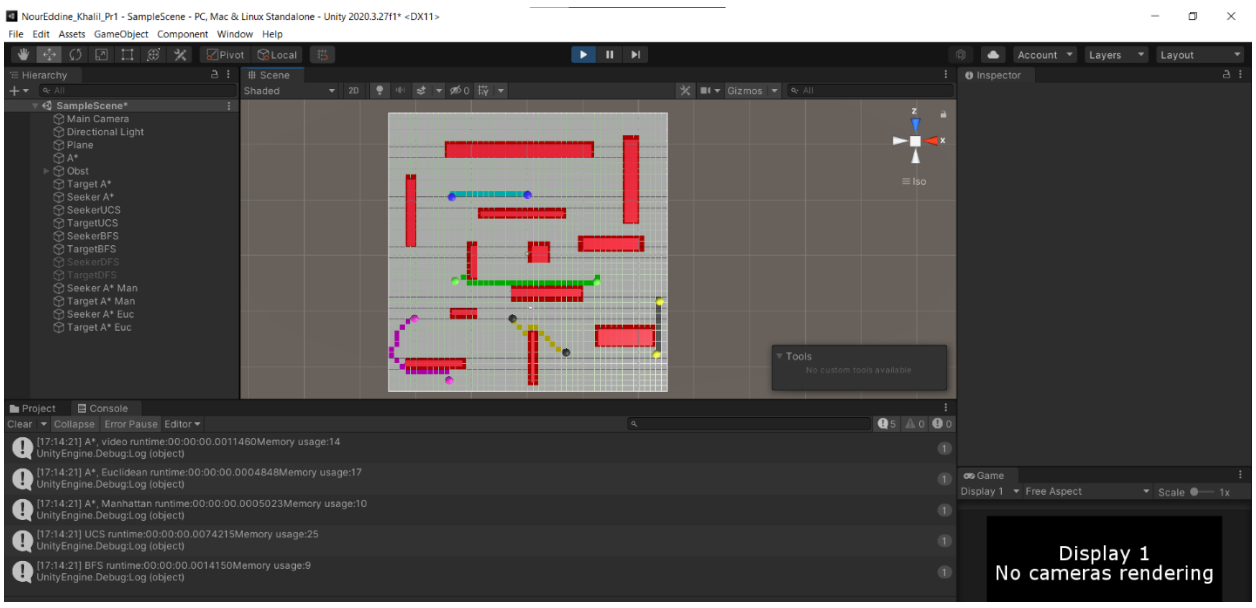

Figure X

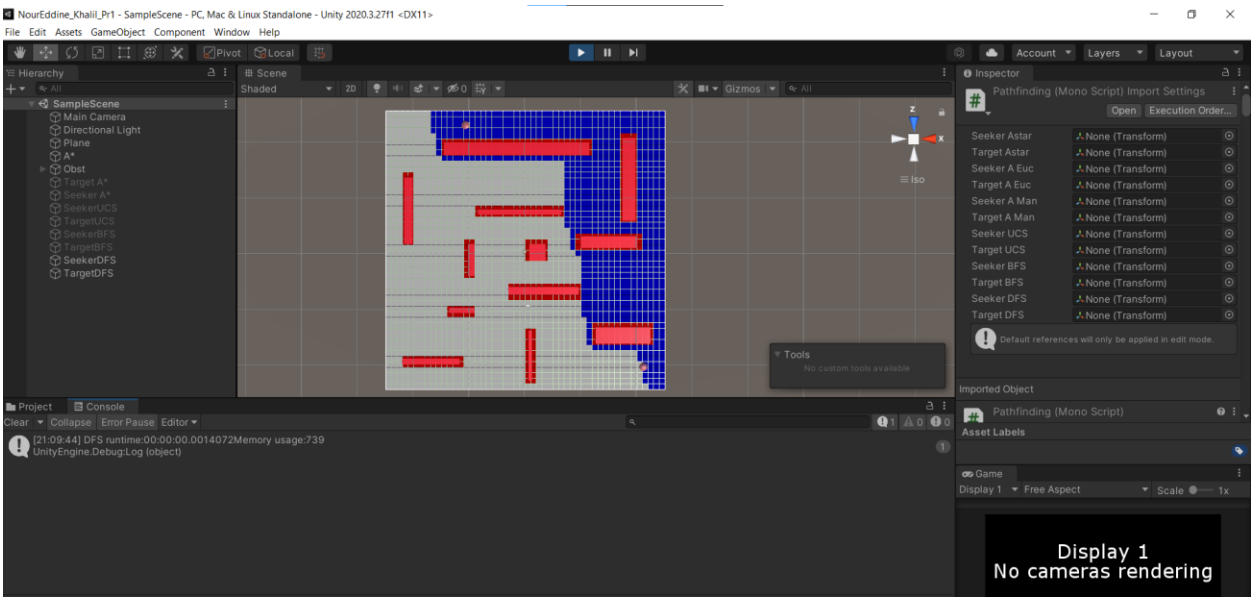

Figure Y

- In the figures X and Y, all the functions were separately run to make sure that they are working.

NB: In the following illustrations, the source node is the one at the top.

I. Comparing A* algorithm with different heuristics:

- A* (video heuristic): Cyan
- A* (Manhattan heuristic): Grey
- A* (Euclidean heuristic): Magenta

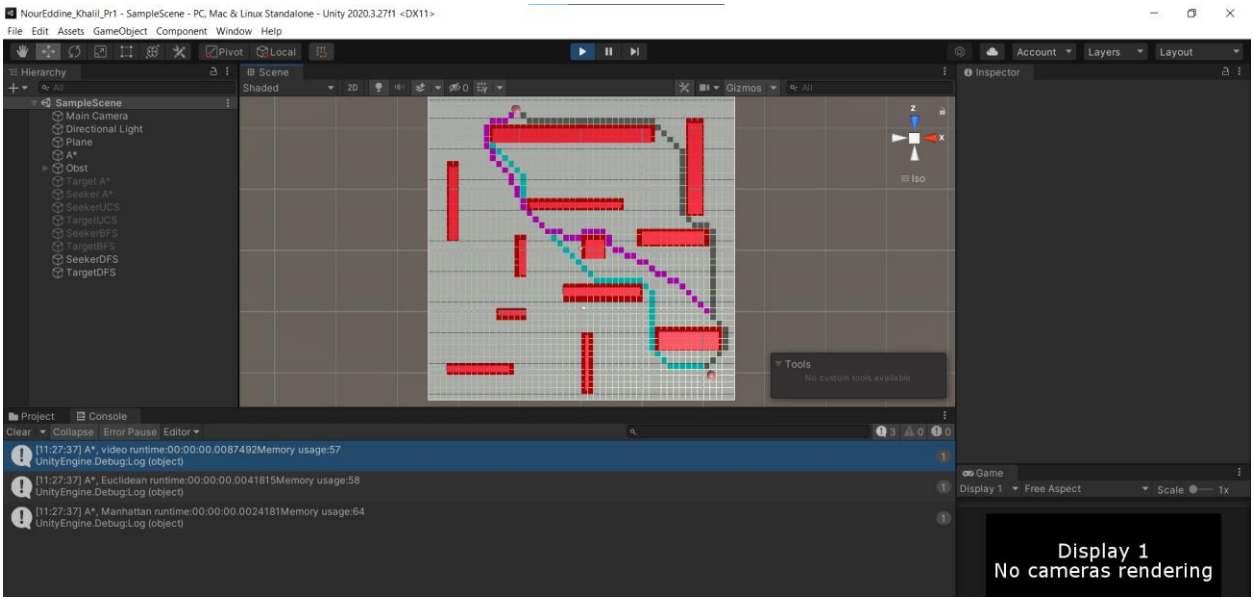

Figure 1

In this figure we can see that the node and the target are relatively far from each other. The A* with the video heuristic and the same algorithm with the Euclidean heuristic do roughly explore the same number of nodes (57/58) hence we can see that in these settings, the Euclidean heuristic was timely efficient compared to the first one (around the half of the time, after running them for different times).

As for the Manhattan heuristic, it did give much efficient results, compared with the other heuristics, when it comes to the time spent to reach the goal, however, the algorithm in this case did explore few nodes than the first two scenarios.

II. Comparing A* (different heuristics), UCS and BFS:

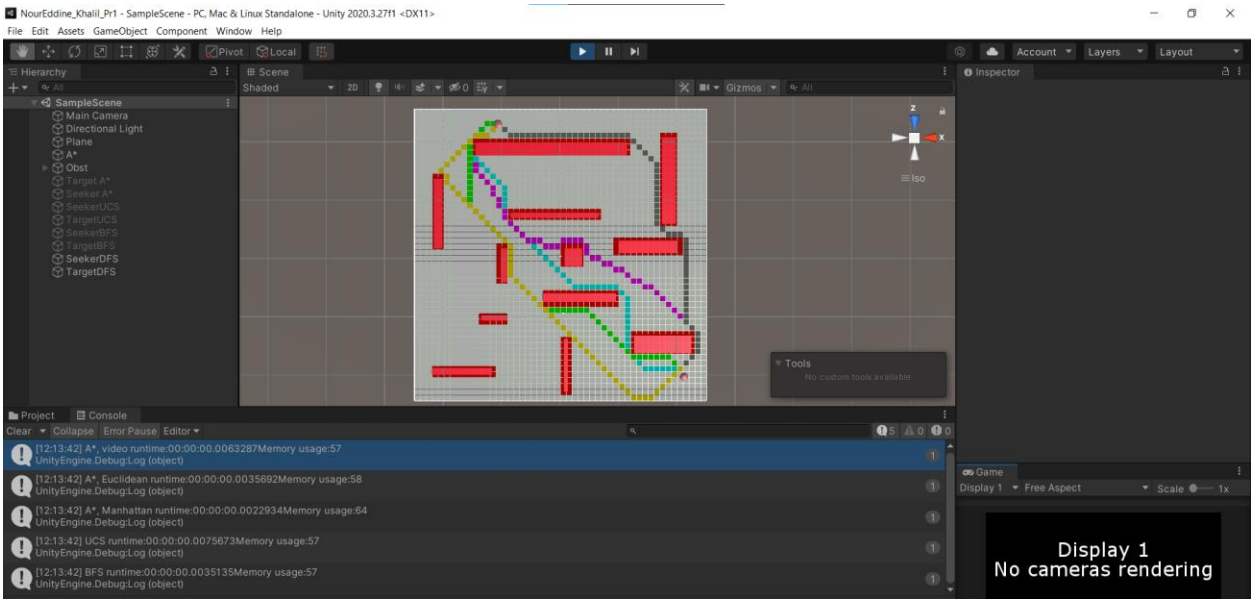

Figure 2

In this figure, we can see that the Uniform-Cost Search (UCS) and Breadth First Search (BFS) algorithm has in this case explored relatively same nodes as the A* algorithm. The UCS is an uninformed search algorithm which expands the nodes with the least cost, and this can explain the fact that it takes more time than the other mentioned algorithms. As for the BFS, it did good in this situation even though it is an uninformed search. BFS expands the children node with the lowest path cost so far and keeps going until the target node is generated.

III. Comparing A* (different heuristics), UCS, BFS and DFS:

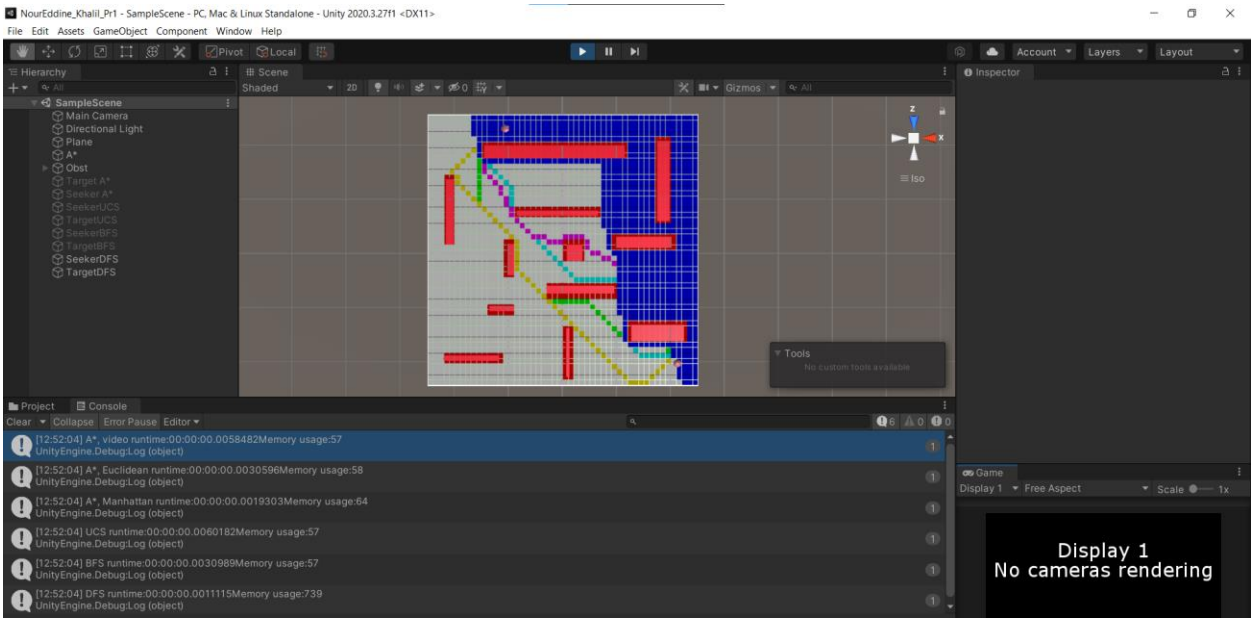

Figure 3

As we can see in figure 3, after running DFS, BFS, UCS, and A* with 3 different heuristics, we can clearly see that DFS is very timely efficient. Hence, DFS does explore 13 times the average visited nodes by other algorithms. Meaning, the space complexity is higher than others while it is still faster, and that is due to the fact that DFS does explore a given node and go ahead and explore its child and the same thing happens until it reached the leaf and then it backtracks. In addition, the A* algorithms with the Manhattan heuristic also were timely efficient compared to the rest of algorithms (except DFS), however, it did explore fewer nodes than they did.

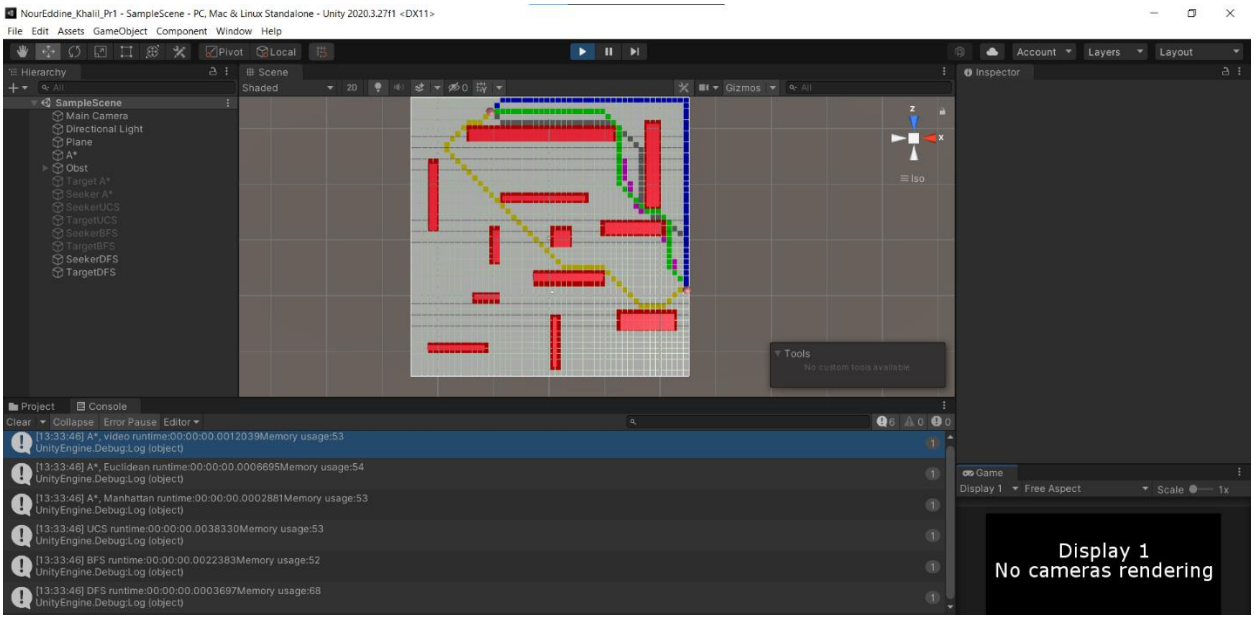

Figure 4

In this scenario, Figure 4, we can see the A* algorithm with the Manhattan heuristic. The distance by this heuristic is computed by counting the number of moves along the grid that each node is displaced from its goal position and summing these values over all tiles. The DFS is second efficient algorithm to find the path, and this is thanks to the fact that while it was exploring the children of the children it reaches the goal. If the goal was on the other side (Figure 5) we can see that it will not be the case. The nodes explored by the DFS were slightly higher than the average nodes explored by other algorithms.

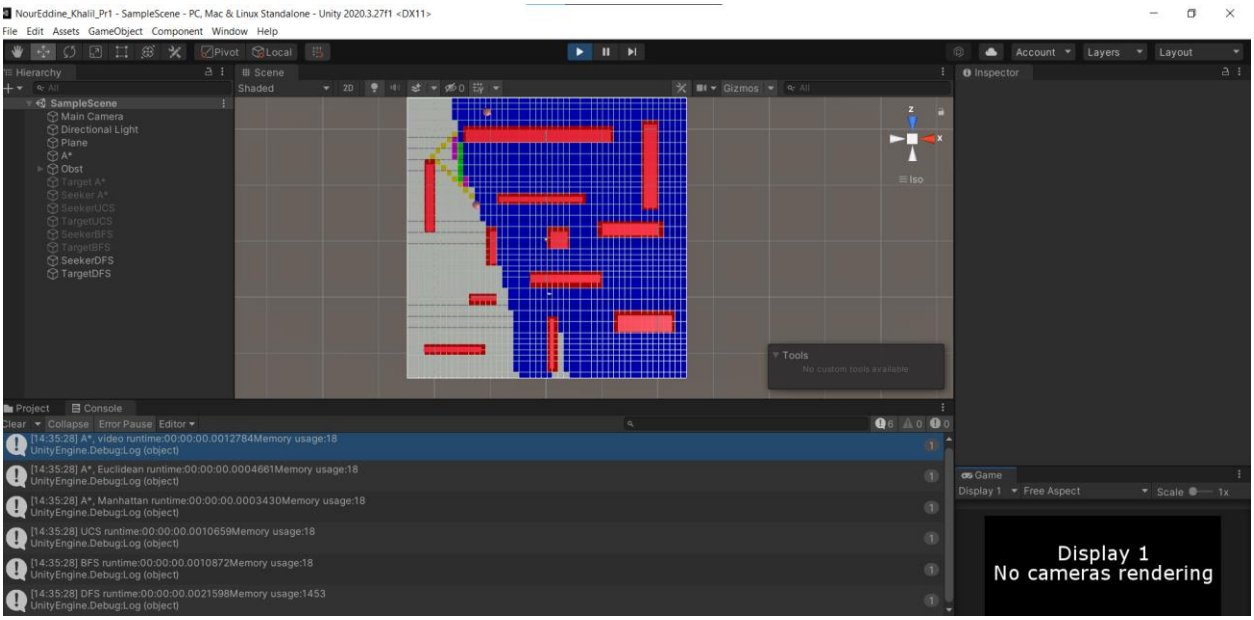

Figure 5

DFS is not efficient in this scenario. It did start exploring the nodes and their children, but it did not reach the goal because it had to explore many nodes till the leaves, which can be explained by the high number of visited nodes and also the time consumed. Other algorithms in this case did very well and again A* with the Manhattan heuristic did win.

IV. Comparing UCS DFS:

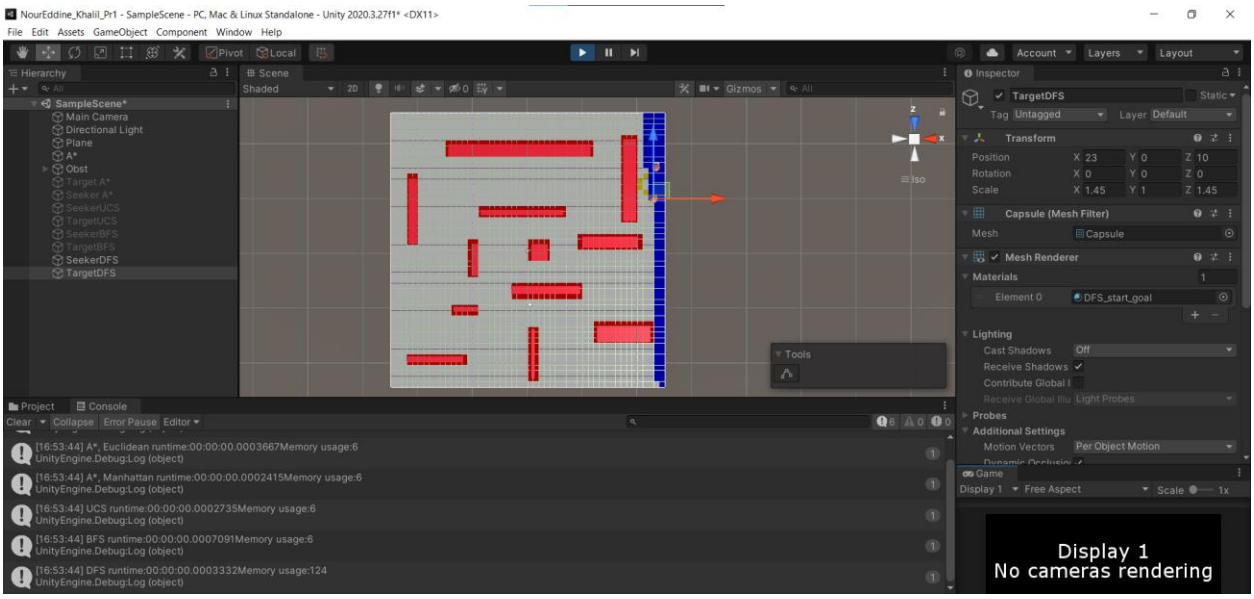

Figure 6

For the test cases that we had before, DFS was always timely efficient, even more than UCS. However, in this case we can clearly see that UCS found the path in less time with less explored nodes than DFS.

Conclusion

After analyzing different scenarios, which were given by different positions of the start / target nodes I noticed that the A* algorithm which is an informed search algorithm can mostly give better results if the heuristic was good enough. The heuristics, which are some educated guesses, can estimate the distance between a current node to the goal node. I noticed also that different heuristics give different path, and that is affected also by the position of the unwalkable nodes (obstacles).

Uninformed search algorithms, DFS, BFS, and UCS also can give good results, but it is not really guaranteed to be optimal all the time. It depends on the positions of the start state and the end state. Compared with informed search algorithms, A* in this case, these algorithms usually take more time to reach the goal and in the case of DFS, it can find be timely efficient hence, it does explore a higher number of nodes.